COMPUTING THE TREE NUMBER OF A CUT-OUTERPLANAR GRAPH

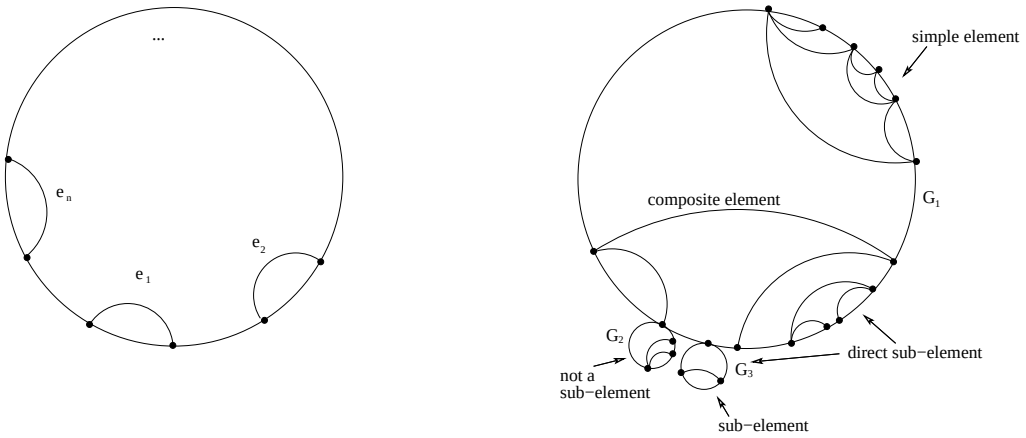
NATALIA VANETIK

Abstract. While the notion of arboricity of a graph is well-known in graph theory, very few results are dedicated to the minimal number of trees covering the edges of a graph, called the tree number of a graph. In this paper we propose a method for computing in polynomial time the tree number of a subclass of planar graphs. The subclass in question includes but is not limited to outerplanar graphs; the difference between arboricity and tree number for graphs in this class can be arbitrarily big.

1. Introduction

A cover of an undirected graph is a partition of its edge set. A cover of a graph is called a *tree cover* if all graphs of the partition are trees. A tree cover of a graph having minimum size is called *minimal tree cover*. A *tree number* of a graph G, denoted $\tau(G)$, is the size of its minimal tree cover. We call a graph a *pseudotree* if it contains exactly one cycle. A cover consisting of acyclic (but not necessarily connected) graphs is called a *forest cover*. The *arboricity* of G is the minimum number of forests in a forest cover of G. The well-known formula for arboricity of graph was given by Nash-Williams in [Nash-Williams '64]. The arboricity provides a lower bound for the tree number. Unfortunately, a general formula for the tree number of a graph is not available although some results are known. The tree number of maximal planar graphs was shown to be at most 3 by Kampen in [Kampen '76]. Ringel proved that maximal bipartite planar graphs have tree number 2 in [Ringel '93]. The nontrivial upper bound for the tree number was obtained by Chung in [Chung '78]. Truszczynsky considered upper bounds in relation to the girth of the graph in [Truszczynsky '88]. Ringel et al. have studied the tree number of regular graphs in [Ringel et al. '97]. Lladó and Lopez have proved a non-trivial bound for the tree number depending on the minimum degree in [Lladó and Lopez '04].

A planar graph is a graph that can be embedded in a plane; a graph is outerplanar if it has an embedding in the plane where the vertices lie on a fixed cycle and the edges lie inside the cycle and do not intersect. Every outerplanar graph is planar, but the converse is not true: K_4 is planar but not outerplanar. We introduce the class of *cut-outerplanar graphs*, that contains the class of outerplanar graphs: such a graph is built from outerplanar graphs connected by cut-nodes in a tree-like fashion. This class is nontrivial, because the difference between graph's arboricity and its tree number can be arbitrarily big (an example in Figure 1(a) shows an outerplanar graph with tree number $n \in \mathbb{N}$ and arboricity 2). Additionally, graphs in this class can have arbitrarily many cut-nodes that usually make the computation of the tree number difficult. We prove that the tree number of cut-outerplanar graph can computed in polynomial time. Section 2 contains definitions used throughout the paper, Sections 3



(a) Graph with tree number n and arboricity 2

(b) Elements and sub−elements

FIGURE 1. Cut-outerplanar graph and its elements.

and 3.1 contain theoretical results necessary for the correctness of the suggested algorithm, and Section 4 contains the algorithm itself and its complexity analysis.

2. DEFINITIONS

Let $G = (V, E)$ be a simple bridgeless graph and let $T = (V_T, E_T)$ be tree describing the node cut structure of G. Each member of V_T is a 2-connected component of G, and two members $G_i, G_j \in V_T$ are connected by an edge $(u, v) \in E_T$ in T if G_i and G_j share a node, denoted c_{ij} (a cut-node of G). We select an arbitrary root of T and provide the edges of T with a direction, thus effectively turning T into an out-tree. We use notation $G_i \prec G_j$ to state that a directed path from G_j to G_i exists in T. If every member of $V_T := \{G_1, ..., G_N\}$ is an outerplanar graph, G is a cut-outerplanar graph.

In this paper, we fix embeddings of $G_1, ..., G_N$ with outer cycles $C_1, ..., C_n$ and provide each C_i with a direction for a given cut-outerplanar graph G. Then a subpath of C_i between nodes u and v, denoted uC_iv, is unambiguously defined. We denote $C := \bigcup_{i=1}^{N} C_i$ and call an edge in E a *C-edge* if it lies on C and *$\overline{C}$-edge* otherwise.

For two nodes $x, y \in C_i$, $\widehat{x, y}$ denotes a subgraph of G that includes all the edges with both ends in xC_iy and all subtrees of T rooted in G_j for which $G_i \prec G_j$ and $c_{ij} \in xC_iy$. If $e = (x, y)$ is an edge, we may write $\widehat{e}$ instead of $\widehat{x, y}$. When $x = y$, $\widehat{x} := \widehat{x, y}$ denotes a subtree T' of T attached to G_i in x, if exists, that is larger than G_i w.r.t. the $\prec$ order. A subgraph L of G is called an *element of* G if $\delta(L) \subseteq C$. There exist nodes $u, v \in V$ for which $L = \widehat{u, v}$, called the end nodes of L (it is possible that $u = v$). G can be viewed as a set of elements connected by C_r-paths, where G_r is the root of T.

An element L_1 is a *sub-element* of an element L_2, denoted $L_1 \subset L_2$, if the edges of L_1 form a subset of edges of L_2, and L_1 and L_2 are connected only by C-paths. An element L_1 is *direct sub-element* of an element L_2 if there exists no element M with $L_1 \subset M \subset L_2$. An element without sub-elements is

called *simple*, and an element with sub-elements is called *composite*. Figure 1(b) contains examples of elements and sub-elements for a cut-outerplanar graph with tree structure $\{G_1 \prec G_2, G_1 \prec G_3\}$.

Direction of each C_i defines a natural order on $\overline{C}$-edges with both ends in C_i. For edges (x, y) and (x', y') with $x, y, x', y' \in C_i$ and $x'C_iy' \subset xC_iy$, we say that (x, y) is a *super-edge* of (x', y') and (x', y') is a *sub-edge* of (x, y). An $\overline{C}$-edge $e = (x, y) \in L$ with $x, y \in C_i$ has *level* 0 *in* L if and only if it has no sub-edge. If e does not have level 0, its *level* is the maximum level of its sub-edges plus 1.

A node x with $d(x) = 2$ is called *fit* in an element L if there exists a tree cover of L where x is covered by two distinct trees and *unfit* otherwise. A minimal tree cover $\mathcal{T}$ of an element L with end nodes u and v is called *proper* if there exist two trees $T_1, T_2 \in \mathcal{T}$ where $u \in T_1$ and $v \in T_2$. Note that u and v may be traversed by other trees from $\mathcal{T}$ as well.

Our goal is compute the tree number for every element of the cut-outerplanar graph G separately and then show how to combine these numbers to get the tree number of G.

3. Tree number of a simple element

Let us first note the following trivial general fact.

$$\tau(L) \geq 2 \text{ for any element } L \text{ of } G, \tag{3.1}$$

because, an element of G always contains a cycle because it either contains an $\overline{C}$-edge or is a simple cycle C_i.

Let us observe a level 1 edge $e = (s, t)$ in an element of G. Then the sub-edges of e have level 0. We refer to these edges as *indifferent set*. Edges $e_1, ..., e_n$ in indifferent set form at most two paths, where each of the paths is incident to either s or t or both by the definition of an element.

Claim 3.1 *Let L be a simple element with no indifferent set. Then $\tau(L) = 2$.*

Proof. Since L contains no indifferent set, all of its $\overline{C}$-edges have level 0. Let L contain the edges of subgraphs $G_{i_1}, ..., G_{i_m}$, with G_{i_1} being the lowest w.r.t. the $\prec$ order. We prove by induction in m that there exists a two-tree cover of L where

$$\text{each cut-node and node incident to an } \overline{C}\text{-edge is traversed by two trees.} \tag{3.2}$$

When $m = 1$, the $\overline{C}$-edges of L form a path with ends in end nodes of L, denoted u and v. If $u \neq v$, $\{L \setminus C, L \cap C\}$ is the required tree cover (see Figure 2(a)). If $u = v$ and L is a simple cycle, we simply separate it into two paths in any degree two node. If, however, $\overline{C}$-edges of L form a cycle, we denote the $\overline{C}$-edges of L by $(a_1, a_2), (a_2, a_3), ..., (a_b, a_1)$. Let $1 \leq j < b$ and x, y be degree two nodes lying on $a_{j-1}C_{i_m}a_j$ and $a_jC_{i_m}a_{j+1}$ respectively. We set $T_1 := L \cap C - (a_j, a_{j+1}) + xC_{i_m}a_j + yC_{i_m}a_{j+1}$ and $T_2 := L \setminus C + (a_j, a_{j+1}) - xC_{i_m}a_j - yC_{i_m}a_{j+1}$ and get the desired tree cover (see Figure 2(b)).

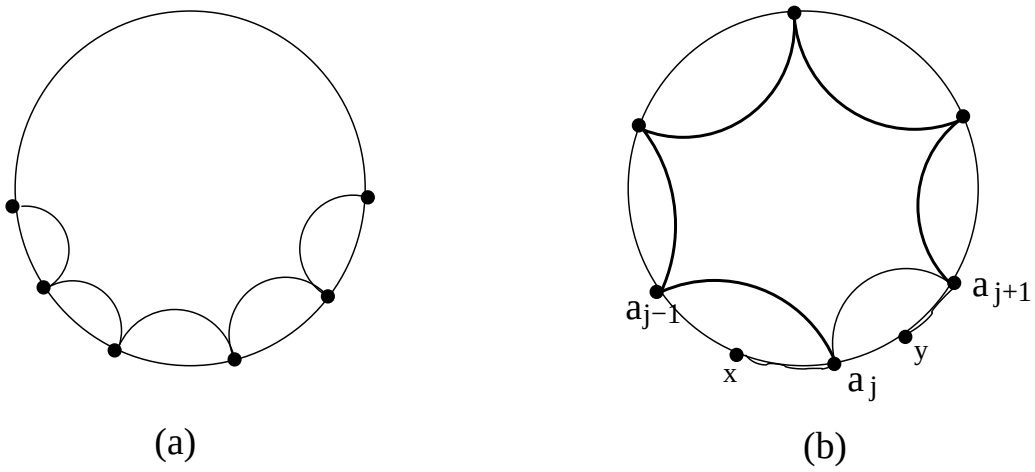


FIGURE 2. Claim 3.1, case $m = 1$.

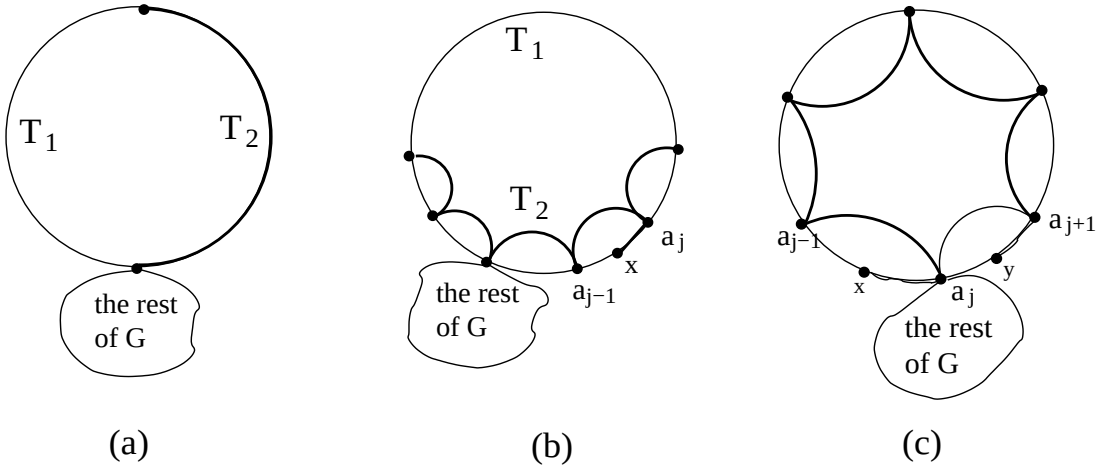


FIGURE 3. Claim 3.1, case $m > 1$.

Otherwise, let G_{i_m} be the largest among $G_{i_1}, ..., G_{i_m}$ w.r.t. the $\prec$ order. Then $L \setminus G_{i_m}$ is also a simple element for which a tree cover $\{T_1, T_2\}$ satisfying the above conditions exists. Let us denote the cut node by which G_{i_m} connects to the rest of L by c_{i_m}. As L is simple, G_{i_m} is either a simple cycle (see Figure 3(a)) or it contains a path $(a_1, a_2), (a_2, a_3), ..., (a_{b-1}, a_b)$ of $\overline{C}$-edges where $a_j = c_{i_m}$ for some $1 \leq j \leq b$ (see Figure 3 (b) and (c)). In the former case we break C_{i_m} in two in some degree two node and add the resulting simple paths to T_1 and T_2. In the latter case, we set $T_1 := T_1 + C_{i_m}$ and $T_2 := T_2 + G_{i_m} \setminus C_{i_m}$. Let $1 \leq j \leq b$ and x be a degree two node lying on $a_{j-1}C_{i_m}a_j$. If T_2 contains no cycle, i.e. $a_1 \neq a_b$, we transfer the segment $xC_{i_m}a_j$ from T_1 to T_2. This way, a cycle in T_1 is eliminated and $\{T_1, T_2\}$ is the required tree cover for L. If T_2 contains a cycle, i.e. $a_1 = a_b$, then $b > 2$ as G is not a multigraph. We perform here the same operation on T_1 and T_2 as in case $m = 1$ (see Figure 3(c)), and obtain the required tree cover. □

Claim 3.2 *Let L be a simple element of G with end nodes u, v and with an indifferent set $e_1, ..., e_n$. Let L' be an element obtained from L by removing $e_1, ..., e_n$. Then $\tau(L) = \tau(L') = 2$.*

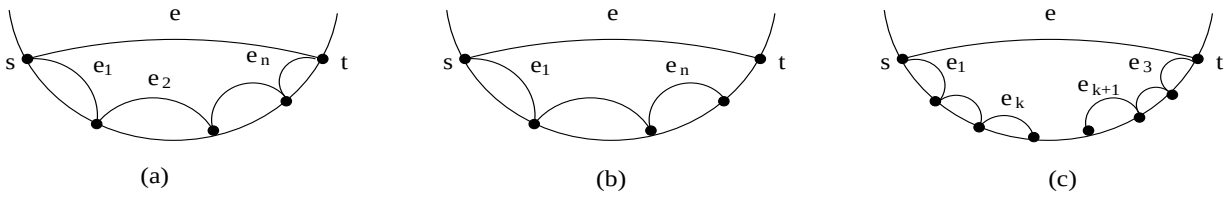


FIGURE 4. Indifferent set.

Proof. Let us observe a counterexample to the claim with L containing minimal number of edges and let $e_1 = (v_1, u_1), ..., e_n = (v_n, u_n)$. Let $e = (s, t) \in L$ be the super-edge of level 1 for $e_1, ..., e_n$. Then $\tau(L') = 2$ and a tree cover $\mathcal{T}' = \{T_1, T_2\}$ of L' exists. Since sC_it, e is a cycle, it is covered by two trees in $\mathcal{T}'$ and w.l.o.g. $(s, t) \in T_1$.

Since L is simple, w.l.o.g. there exist three possibilities of $e_1, ..., e_n$ location (see Figure 4(a), (b) and (c)):

(1) $s = v_1$, $t = u_n$ and $e_1, ..., e_n$ is a path,
(2) $s = v_1$, $t \neq u_n$ and $e_1, ..., e_n$ is a path,
(3) $s = v_1$, $t = u_n$ and $e_1, ..., e_k$, $e_{k+1}, ..., e_n$ are two disjoint paths.

Let us first assume that $sC_it \in T_2$. We then set $\mathcal{T} := \{T_1 \cup e_1 \cup ... \cup e_n, T_2\}$. For cases 2-3 $\mathcal{T}$ is the required tree cover for L. In case 1, there exists a degree 2 node w lying on $v_nC_iu_n$ as G is not a multigraph. Then $\mathcal{T} := \{T_1 + e_1 + ... + e_{n-1} + v_nC_iw, T_2 - v_nC_iw + e_n\}$ is the required tree cover.

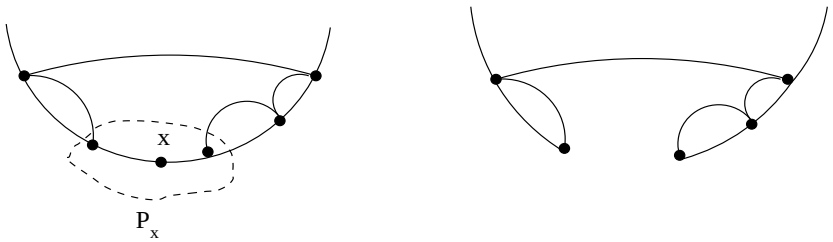
Let us now assume that $sC_ix \in T_1$ and $xC_it \in T_2$ for some degree two node $x \in sC_it$. We set $T_2 := T_2 + sC_ix$ (note that T_2 gains a cycle) and repeat the operations described above. Now we only need to destroy the cycle in the new T_2. Since G is not a multigraph, a degree two node w lying on $v_1C_iu_1$ exists, and by construction $(v_1, u_1) \in T_1$ and $v_1C_iu_1 \in T_2$. We set $T_2 := T_2 - wC_iu_1$ and $T_1 := T_1 + wC_iu_1$ and eliminate the cycle from T_2; we then have a tree cover of size 2 for L. From (3.1) we deduce that $\tau(L) = 2$. □

From Claims 3.1 and 3.2 we know that

(3.3) *the tree number of a simple element is* 2.

3.1. **Fitness in a simple element.** We now describe a method for determining whether or not a node is fit in a simple element L, when other fit or unfit nodes may exist. For a degree two node x, let us call the longest C-path containing x and not incident to $\overline{C}$-edges the *fitness path* of x, denoted by P_x. Note that removing P_x from a simple element leaves the graph cut-outerplanar (see Figure 5.)

Claim 3.3 *Let P_x be a fitness path of a node x. Let $L' := L \setminus P_x$ be a subgraph. Then $\tau(L') = \tau(L)$ if and only if x is fit.*



(a) L before P_x is removed. (b) L after P_x is removed.

FIGURE 5. Removing a fitness path.

Proof. Let us assume that x is fit. Then there exists a tree cover $\mathcal{T} = \{T_1, T_2\}$ of L where $P_x = P_1 x P_2$ and w.l.o.g. $P_1 \in T_1$, $P_2 \in T_2$. Setting $T_1 := T_1 - P_1$ and $T_2 := T_2 - P_2$ gives us a tree cover of size 2 for L'.

Let us assume now that $\tau(L') = 2$ for an unfit x. We select L to be minimal in the number of $\overline{C}$-edges. Equation (3.3) and $\tau(L') > 2$ imply that L' consists of two simple elements, possibly connected by a simple path. Then $\tau(L') \geq 3$ because the tree number of each simple element is 2 and these elements can share at most one tree in a minimal tree cover of L'.

Let us assume first that L consists of two simple elements, denoted L_1 and L_2, connected by a cut-node y and w.l.o.g. $P_x \in L_1$. Removing P_x from L_1 disconnects L_1 as an element if and only if removing P_x disconnects L as an element. Therefore, x is unfit in L_1. However, separating the two-tree cover of L' at y gives us two-tree covers for both L_1 and L_2 - a contradiction.

The remaining case is when L consists of a simple element M with end nodes u, v and an edge (u', v) with u', u, v appearing on C in that order. If x is unfit in M, removing P_x separates M and thus separates L as the edge (u', v) can be incident to only one of the simple elements obtained from separating M. Then $\tau(L') > 2$ as required. If x is fit in M, there exists a proper tree cover $\mathcal{T} = \{T_1, T_2\}$ for M where x is traversed by two trees. If w.l.o.g. v is traversed by T_1 and T_2 and u is traversed by T_1, we set $T_2 := T_2 + (u', v)$ and $T_1 := T_1 + u'Cu$. If u is traversed by T_1 and T_2 and v is traversed by T_1, we set $T_1 := T_1 + (u', v)$ and $T_2 := T_2 + u'Cu$. In both cases we obtain a tree cover of size 2 for L where x is traversed by two trees - a contradiction. □

Corollary 3.4 *Let x_1 and x_2 be a degree two nodes in a simple element L with fitness paths $P_{x_1} = P_{x_2}$. Then x_1 and x_2 are fit in L simultaneously only if L contains precisely one $\overline{C}$-edge.*

Proof. Let $\mathcal{T} = \{T_1, T_2\}$ be a tree cover of L with T_1, T_2 both traversing x_1 and x_2. Then, assuming x_1 precedes x_2 on C, $T_1 = x_1 C x_2$. This is only possible if every cycle of L contains $x_1 C x_2$, which is exactly the case when L contains only one $\overline{C}$-edge. □

Corollary 3.5 *Let x_1 and x_2 be fit nodes in a simple element L with fitness paths P_{x_1} and P_{x_2}. Then x_1 and x_2 are unfit in L simultaneously only if $P_{x_1} = P_{x_2}$.*

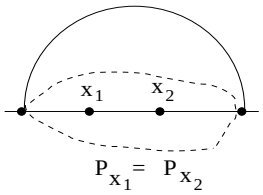


FIGURE 6. Two nodes sharing a fitness path in a simple element.

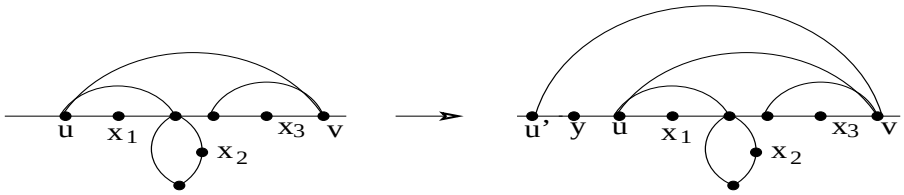


FIGURE 7. Existence of proper tree cover.

Proof. Let us assume that $P_{x_1} \neq P_{x_2}$ while x_1 and x_2 are unfit in L. Let us denote $L' := L \backslash \{P_{x_1} \cup P_{x_2}\}$. Then $\tau(L') > 2$, because removing fitness path of simultaneously fit nodes preserves the two-tree cover of L where every such node is covered by two trees. However, if removing a fitness path after others turns L into a non-element, the tree cover of resulting graph has to be bigger than 2. Then $\tau(L') > 2$ and L' is not a single simple element.

Let L' consist of simple elements $L'_1, ..., L'_m$, possibly connected by simple paths (some of L'_i may be trivial). Adding P_{x_1} to L' turns it into a single simple element, meaning that $m = 2$ and both L'_1, L'_2 are nontrivial (attaching a simple path to L'_i does not change the tree number of L'_j). The same is true for P_{x_2}. Then both P_{x_1} and P_{x_2} connect L'_1 and L'_2 in L. Since L is simple, L'_1 cannot be a sub-element of L'_2 and vice versa, thus exactly one C-path connects L'_1 and L'_2 in L. But then P_{x_1} and P_{x_2} have a common segment as they are subpaths of C, which is impossible unless $P_{x_1} = P_{x_2}$ (see Figure 6). □

3.2. **Proper tree cover.** The goal of this section is to determine when a simple element admits a proper tree cover subject to additional conditions.

Claim 3.6 *Let L be a simple with end nodes u, v and with degree two nodes $x_1, ..., x_n$ that are fit in L simultaneously. Let L' be an element obtained from L by adding a (u', v)-edge and a degree two node y where u', y, u, v appear on C in that order (see Figure 7). Then there exists a proper tree cover $\mathcal{T}$ of L where all $x_1, ..., x_n$ are traversed by two trees if and only if nodes $x_1, ..., x_n, y$ are fit in L' simultaneously.*

Proof. If $x_1, ..., x_n, y$ are fit in L' simultaneously, there exists a minimal tree cover $\mathcal{T}' = \{T_1, T_2\}$ of L' where $x_1, ..., x_n$ are traversed by T_1 and T_2 and w.l.o.g. $(u', v), (u', y) \in T_1$ and $(y, u) \in T_2$. Setting $T_1 := T_1 - (u', v) - (u', y)$ and $T_2 := T_2 - (y, u)$ gives us the desired proper cover. Suppose that nodes $x_1, ..., x_n, y$ are not fit in L' simultaneously but a proper tree cover $\mathcal{T}$ of L where all $x_1, ..., x_n$ are

FIGURE 8. Cases of Claim 4.1.

traversed by two trees and $u \in T_1 \in \mathcal{T}$ and $v \in T_2 \in \mathcal{T}$ exists. Then setting $T_1 := T_1 + (y, u)$ and $T_2 := T_2 - (u', v) - (u', y)$ gives us a minimal tree cover for L' where all $x_1, ..., x_n, y$ are traversed by two trees each - a contradiction. □

Claim 3.6 and Corollary 3.5 together imply:

Corollary 3.7 *In a simple element L with end nodes u, v and with simultaneously fit nodes $x_1, ..., x_n$. Then there exists a proper tree cover $\mathcal{T}$ of L where all $x_1, ..., x_n$ are traversed by two trees if and only if no two fitness paths of $x_1, ..., x_n$ are the same.* □

The above corollary allows us to determine when a simple element admits a proper tree cover while some degree two nodes inside of that element may be required to be fit. We use this result later to determine the existence of a proper tree cover for a composite element.

4. Tree number of a composite element

In this section we describe how the tree number of a composite element can be computed iteratively from the tree numbers of its direct sub-elements.

Let L be an element of G and M its direct sub-element. We denote by $L - M$ an element obtained by *contracting* M into a degree two node m. If $M_1, ..., M_n$ are all direct sub-elements of L, then contracting $M_1, ..., M_n$ into degree two nodes $m_1, ..., m_n$ brings us to a simple element $L - \bigcup_{i=1}^{n} M_i$. Node m_i will be called a *contraction node* of M_i. We can then make the following claims about the tree number of G.

Claim 4.1 $\tau(L) \geq \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - 2n$.

Proof. We prove the claim by induction in n. Let us observe a minimal tree cover $\mathcal{T}$ of L. We denote by e_1 and e_2 the two C-edges connecting M_1 to the rest of L. If $e_1 \in T_1 \in \mathcal{T}$ and $e_1 \in T_2 \in \mathcal{T} \setminus T_2$ (see Figure 8(a)), we immediately get $\tau(L) \geq \tau(L - M_1) + \tau(M_1) - 2$. If $e_1, e_2 \in T_1 \in \mathcal{T}$ (see Figure 8(b)), L admits a cover where one cover member is a pseudotree and others are trees. The size of such a cover is at least $\tau(L) - 1$ (otherwise, breaking the cycle of a pseudotree by creating a new tree from one of its cycle's edges), and at least $\tau(M_1) - 1$ trees cover M_1 in $\mathcal{T}$. Then again $\tau(L) \geq \tau(L - M_1) + \tau(M_1) - 2$. Induction assumption gives us $\tau(L - M_1) \geq \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=2}^{n} \tau(M_i) - 2(n-1)$, thus $\tau(L) =$

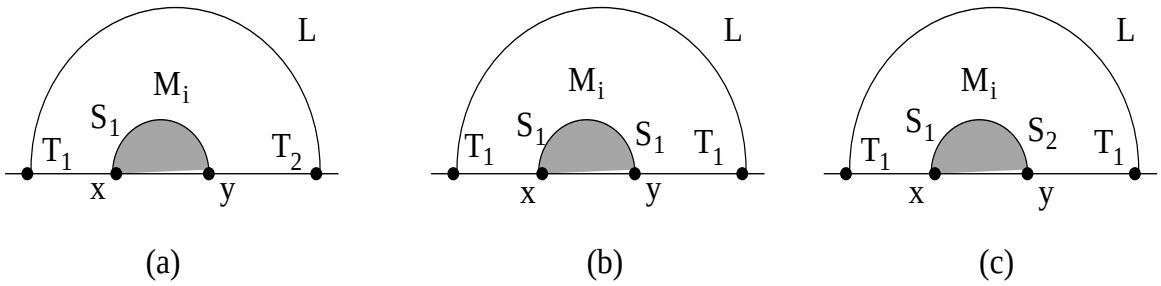


FIGURE 9. Cases of Claim 4.2.

$\tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - 2n$ as required. □

Claim 4.2 $\tau(L) \leq \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n$.

Proof. We prove the claim by induction in n. Let us observe minimal tree covers T_{L-M_1} of M_1 and T_{M_1} of M_1. We denote by x and y the end nodes of M_1, and by e_x and e_y the C-edges of $L - M_1$ incident to the node m_1 obtained by compressing M_1. The first case is when e_x and e_y are traversed by two different trees of T_{L-M_1}, denoted T_1 and T_2 (see Figure 9(a)). Then taking a tree $S_1 \in T_{M_1}$ traversing x, we set $T_1 := T_1 + S_1$ and obtain a tree cover of size $\tau(L - M_1) + \tau(M_1) - 1$ for L. The second case is when e_x and e_y by the same tree $T_1 \in T_{L-M_1}$. If there exists a tree $S_1 \in T_{M_1}$ traversing both x and y (see Figure 9(b)), we set $T_1 := T_1 + S_1$ and obtain a tree cover of size $\tau(L - M_1) + \tau(M_1) - 1$ for L. Otherwise, there exist trees $S_1, S_2 \in T_{M_1}$ traversing x and y respectively (see Figure 9(c)). We break T_1 into two trees T_1' and T_1'', one covering e_x and another e_y, and set $T_1' := T_1' + S_1$ and $T_2'' := T_1'' + S_2$. The resulting tree cover of L has the size $\tau(L - M_1) + 1 + \tau(M_1) - 2 = \tau(L - M_1) + \tau(M_1) - 1$. Then by induction we have $\tau(L) \leq \tau(L - M_1) + \tau(M_1) - 1 \leq \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n$, as required. □

Claims 4.1 and 4.2 together imply that for all $i \in \{1, ..., n\}$

$$\tau(L - M_i) + \tau(M_i) - 2 \leq \tau(L) \leq \tau(L - M_i) + \tau(M_i) - 1. \tag{4.4}$$

We will also need the following property of L.

Claim 4.3 *Let $\tau(L) = \tau(L - M_i) + \tau(M_i) - 2$ for some $i \in \{1, ..., n\}$. Then m_i is fit in $L - M_i$ and M_i admits a proper tree cover.*

Proof. Let us assume that $\tau(L) = \tau(L - M_i) + \tau(M_i) - 2$ for some $i \in \{1, ..., n\}$ and let $\mathcal{T}$ be a minimal tree cover of L. If the C-edges of $L \setminus M_i$ incident to M_i are traversed by two different trees $T_1, T_2 \in \mathcal{T}$, then contracting M_i into m_i gives us a tree cover for $L - M_i$, built from $\mathcal{T}$, where m_i is traversed by two different trees and thus is fit. Deleting $L \setminus M_i$ from L gives us a tree cover of M_i with end nodes traversed by T_1 and T_2 - a contradiction.

If the C-edges of $L \setminus M_i$ incident to M_i are traversed by a single tree $T_1 \in \mathcal{T}$, M_i is covered by $\tau(M) - 1$ trees in $\mathcal{T}$. Since no tree cover of M_i can contain less than $\tau(M_i)$ trees, breaking T_1 at the end nodes

of M_i gives us a proper tree cover of M_i. The cover of $L - M_i$, obtained from $\mathcal{T}$ by contracting M_i, contains precisely one pseudotree traversing m_i and has size $\tau(L - M_i) - 1$. We create a new tree from one of the C-edges incident to m_i and obtain a tree cover of $L - M_i$ where m_i is traversed by two different trees. Then m_i is fit in $L - M_i$. □

Let $\mathcal{T}$ be any minimal tree cover of L and $\mathcal{T}_{M_i}$ be the restriction of $\mathcal{T}$ onto M_i. Then

$$|\mathcal{T}_{M_i}| = \tau(M_i). \tag{4.5}$$

Indeed, if $\tau(L) = \tau(L - M_i) + \tau(M_i) - 2$, $L - M_i$ and M_i can share exactly two trees, thus $\mathcal{T}_i$ contains precisely $\tau(M_i)$ trees. If $\tau(L) = \tau(L - M_i) + \tau(M_i) - 1$, either $L - M_i$ and M_i share exactly one tree in $\mathcal{T}$, and thus $|\mathcal{T}_{M_i}| = \tau(M_i)$, or $L - M_i$ and M_i share two trees and $L - M_i$ is covered by $\tau(L - M_i) + 1$ trees. In this case, again, $|\mathcal{T}_{M_i}| = \tau(M_i)$.

Claim 4.4 *Let us select the largest index set $\{i_1, ..., i_k\} \subseteq \{1, ..., n\}$ so that all $M_{i_1}, ..., M_{i_k}$ admit proper covers and $m_{i_1}, ..., m_{i_k}$ are fit simultaneously in $L - \bigcup_{i=1}^{n} M_i$. Then*
$\tau(L) = \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n - k$.

Proof. We prove the claim by induction in k. If $k = 0$, Claims 4.3, 4.2 and 4.1 ensure that $\tau(L) = \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - 2n$. If $k > 0$, we denote by $L' := L - M_{i_1}$. By induction assumption $\tau(L') = \tau(L - \bigcup_{i=2}^{n} M_i) + \sum_{i=2}^{n} \tau(M_i) - n - k + 2$, as $m_{i_2}, ..., m_{i_k}$ are fit simultaneously in $L - \bigcup_{i=1}^{n} M_i$. Since all M_{i_1} admit a proper tree covers, and $m_{i_1}, ..., m_{i_k}$ are fit simultaneously in $L - \bigcup_{i=1}^{n} M_i$, we simply construct the required tree cover by joining each two trees covering m_j in a tree cover of $L - \bigcup_{i=1}^{n} M_i$ with the two trees covering the end nodes of each M_{i_j}. For each M_j, $j \notin \{i_1, ..., i_k\}$, construct a tree cover of size $\tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n - k$ as in Claim 4.2. By (4.4) $\tau(L) \in \{\tau(L') + \tau(M_{i_1}) - 1, \tau(L') + \tau(M_{i_1}) - 2\}$. Then by induction assumption $\tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n - k$ is the minimal possible size of for the tree cover of L. □

4.1. **Proper tree cover of a composite element.** In this section we study the existence of a proper tree for a composite element, when properties of the element are known. Here, L denotes a composite element and $M_1, ..., M_n$ denote the direct sub-elements of L. We will call M_i and M_j *parallel* in L if any maximal subset of $M_1, ..., M_n$ with proper tree covers and simultaneously fit contraction nodes in L contains both M_i and M_j. The intuition behind this definition is that only parallel direct sub-elements may affect the existence of a proper tree cover in composite element, as the following claim states.

Claim 4.5 *L admits a proper tree cover unless there exist parallel direct sub-elements M_i and M_j so that m_i and m_j have the same fitness path.*

Proof. If no such pair of sub-elements exists, applying construction of Claim 4.4 to the proper tree cover of $L - \bigcup_{i=1}^{n} M_i$, which exists by Corollary 3.7, gives us the required tree cover. Otherwise, let us observe a proper tree cover $\mathcal{T}$ of L and parallel direct sub-elements M_i and M_j so that m_i and m_j

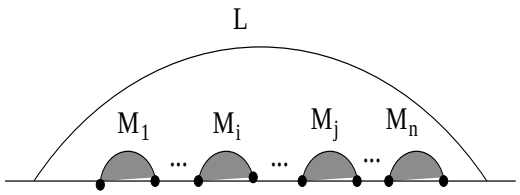


FIGURE 10. Sub-elements in Claim 4.5.

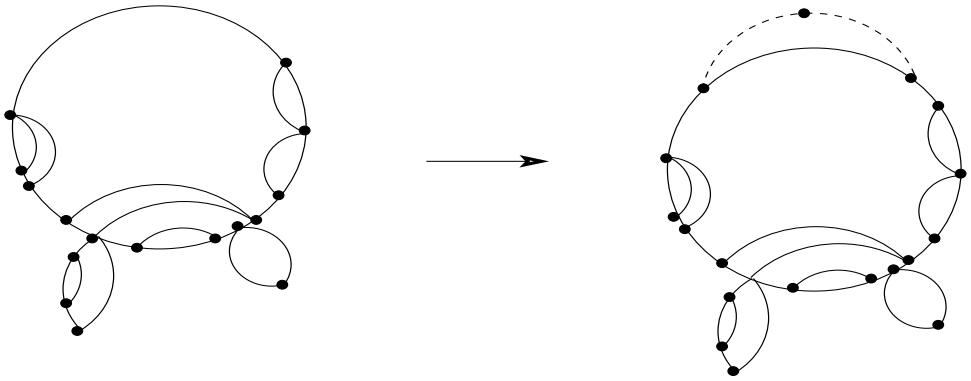


FIGURE 11. Building element from graph.

have the same fitness path $P_{m_i} = P_{m_j}$. Contracting $M_{i_1}, ..., M_{i_k}$ brings us to a proper tree cover of $L' := L - \bigcup_{j=1}^{k} M_{i_j}$, where the nodes $m_{i_1}, ..., m_{i_k}$ are fit simultaneously. In $L - \bigcup_{i=1}^{n} M_i$, by Corollary 3.4 no $m_p \neq m_i, m_j$ is fit and $L - \bigcup_{i=1}^{n} M_i$ contains precisely one $\overline{C}$-edge. We then have

$$\tau(L) = \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n - 2 = \sum_{i=1}^{n} \tau(M_i) - n, \tag{4.6}$$

from Claim 4.4 as $\tau(L - \bigcup_{i=1}^{n} M_i) = 2$ by (3.3). By (4.5) all $M_1, ..., M_n$ are covered by $\tau(M_1), ..., \tau(M_n)$ trees respectively in $\mathcal{T}$. For the equality in (4.6) to hold, every path in L connecting a sub-element M_i to another sub-element M_j has to be covered by a single tree (see Figure 10). This includes the path between the outermost direct sub-elements on C, consisting of the edges incident to the end nodes of L. Then $\mathcal{T}$ cannot be proper - a contradiction. □

5. Computing the tree number

Here, we describe an algorithm for computing the tree number of a cut-outerplanar graph and show that the algorithm's time complexity is polynomial in the size of the input.

First, we notice that an algorithm that computes the tree number of a composite element of a cut-outerplanar graph can compute the tree number of a cut-outerplanar graph. Indeed, a cut-outerplanar graph G can be treated as an element of another cut-outerplanar graph G' connected to G by a simple path as in Figure 11.

Let L be an element (simple or composite) of a cut-outerplanar graph, and let $M_1, ..., M_n$ be its direct sub-elements. We will now compute $\tau(L)$ using following steps.

(1) If $n = 0$, $\tau(L) = 2$ by (3.3) and L always admits a proper tree cover by Corollary 3.7.

(2) If $n > 0$, we compute $\tau(M_1), ..., \tau(M_n)$ by recursive application of the algorithm. We also find subset $\{M_{i_1}, ..., M_{i_k}\} \subseteq \{M_1, ..., M_n\}$ of elements with a proper tree cover.

(3) We contract $M_1, ..., M_n$ into degree two nodes $m_1, ..., m_n$ and obtain a simple element $L - \bigcup_{i=1}^{n} M_i$.

(4) We find the maximum subset $I \subseteq \{1, ..., n\}$ so that each $i \in I$ lies in $\{i_1, ..., i_k\}$ and all m_i, $i \in I$, are fit simultaneously. This is done as follows.

   (a) Observe degree two nodes $m_{i_1}, ..., m_{i_k}$ in a simple element $L - \bigcup_{i=1}^{n} M_i$.

   (b) Select maximal subset $\mathcal{M}$ of $m_{i_1}, ..., m_{i_k}$ so that no two nodes have the same fitness path. Do it by picking one node out of every group with the same fitness path per group.

   (c) If $|\mathcal{M}| = 1$, and there exist two nodes $m', m'' \in \{m_{i_1}, ..., m_{i_k}\}$ with the same fitness path and $L - \bigcup_{i=1}^{n} M_i$ contains precisely one $\overline{C}$-edge, let I contain the indices of m', m''. Otherwise, let I contain the indices of the nodes in $\mathcal{M}$.

   Correctness of this step in ensured by Corollary 3.5.

(5) Claim 4.4 ensures that $\tau(L) = \tau(L - \bigcup_{i=1}^{n} M_i) + \sum_{i=1}^{n} \tau(M_i) - n - |I|$.

Since the element $L - \bigcup_{i=1}^{n} M_i$ is simple, we conclude from (3.3) that $\tau(L) = 2 + \sum_{i=1}^{n} \tau(M_i) - n - |I|$.

It remains to show that complexity of the above algorithm is polynomial. Finding cut-nodes in an outerplanar graph at worst requires deleting each node and checking the connectivity of the remaining graph, which can be done by applying depth-first search and is polynomial in the size of the graph. In general, finding an outerplanar embedding of an outerplanar graph is a polynomial task (see [Bienstock, Monma '90]). Such an embedding has to be found for every 2-connected component of a cut-outerplanar graph.

First, the tree number of each sub-element of L is computed exactly once, thus the total number of these computations does not exceed the number of edges in L. In step 2, existence of a proper tree cover of an element M is determined by its tree number and its sub-elements. Since computation of $\tau(M)$ in step 4 includes this data by Claim 4.5, no extra computation is required. Contraction of step 3 is straightforward and is linear in the size of L. Computation in step 4 requires fitness path computation and a linear scan afterward. Computing fitness paths requires a walk along C between the end nodes of L, which is by itself linear in the number of nodes in L. The complexity of finding the end nodes of L is at worst quadratic in the number of nodes in L. Finally, the complexity step 5 is linear in the number of direct sub-elements of L. Therefore, the total complexity of computing the tree number of a cut-outerplanar graph is polynomial.

[Bienstock, Monma '90] Daniel Bienstock and Clyde L. Monma, *On the complexity of embedding planar graphs to minimize certain distance measures*, Algorithmica, 5(1), pp.93109, 1990.

[Chung '78] F.R.K. Chung, *On partition of graphs into trees*, Discrete Math. 23 (1978), pp.23-30.

[Kampen '76] G.R Kampen, *Orienting planar graphs*, Discrete Math. 14 (1976) 337-341.

[Lladó and Lopez '04] A. Lladó and S.C. Lopez, *Minimum degree and minimum number of edge-disjoint trees*, Discrete Mathematics 275 (1-3), pp. 195-205.

[Nash-Williams '64] C.St.J.A. Nash-Williams, *Decomposition of finite graphs into forests*, J. London Math. Soc. 39, 12 (1964).

[Ringel et al. '97] Gerhard Ringel, Anna S. Lladó and Orio Serra, *On the tree number of regular graphs*, in Proceedings of an international symposium on graphs and combinatorics, Marseille, France, pp. 587-595, 1997.

[Ringel '93] G. Ringel, *Decomposition of maximal bipartite planar graphs into two trees*, J. Graph Theory 17 (1993), pp. 755-758.

[Truszczynsky '88] M. Truszczynsky, *The tree number of a graph with a given girth*, Per. Math. Hungarica 19 (1988) pp. 273-286.